AFRL-RI-RS-TR-2014-263

CONSTRAINT DRIVE GENERATION OF VISION ALGORITHMS ON AN ELASTIC INFRASTRUCTURE

UNIVERSITY OF MARYLAND

OCTOBER 2014

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

AFRL-RI-RS-TR-2014-263 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/ S /	**/ S /**
PATRICK K. McCABE	MICHAEL J. WESSING
Work Unit Manager	Deputy Chief, Information Intelligence
	Systems and Analysis Division
	Information Directorate

REPORT DOCUMENTATION PAGE	***Form Approved*** **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)* OCTOBER 2014	**2. REPORT TYPE** FINAL TECHNICAL REPORT	**3. DATES COVERED** *(From - To)* MAY 2012 – MAY 2014

4. TITLE AND SUBTITLE CONSTRAINT DRIVE GENERATION OF VISION ALGORITHMS ON AN ELASTIC INFRASTRUCTURE	**5a. CONTRACT NUMBER** FA8750-12-C-0180
	5b. GRANT NUMBER N/A
	5c. PROGRAM ELEMENT NUMBER 62305E
6. AUTHOR(S) Brandyn A. White, Larry S. Davis	**5d. PROJECT NUMBER** VMRG
	5e. TASK NUMBER 00
	5f. WORK UNIT NUMBER 09
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Institute for Advanced Computer Studies University of Maryland College Park , MD 20742	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RIED 525 Brooks Road Rome NY 13441-4505	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RI
	11. SPONSOR/MONITOR'S REPORT NUMBER AFRL-RI-RS-TR-2014-263

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In this project a prototype Shopforeman system was developed that composes heterogeneous computer vision algorithms on the fly along with a complementary Nightwatchman system that automatically generates novel concept classifiers for the Shopforeman to use. The Shopforeman provides an interface to the user that allows them to specify the specific problem they are trying to solve and a target operating point. The goal of this project was to explore the design of an automated interface to a heterogeneous collection of computer vision algorithms and complementary approaches to develop concept classifiers and datasets that they use.

15. SUBJECT TERMS
Heterogeneous computer vision algorithms, Nightwatchman, Shopforeman, concept classifiers, Picarus machine learning system

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** **PATRICK K. McCABE**
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	29	**19b. TELEPHONE NUMBER** (***Include area code)*** **315-330-3197**

TABLE OF CONTENTS

List of Figures i

1 Summary 1

2 Introduction 2

3 Methods, Assumptions, and Procedures 7

3.1 Shop Foreman 7

3.1.1 Types 7

3.1.2 Datasets 7

3.1.3 Components 8

3.1.4 Picarus 10

3.2 Cascade Calibration 11

3.2.1 Proposed Approach 11

3.2.2 Cascade Stages 12

3.2.3 Threshold Selection 12

3.2.4 Cascade Selection/Simulation 13

3.2.5 Experiments 14

3.3 Nightwatchman 15

3.3.1 Annotation 16

3.3.2 Active Learning: Batch 16

3.3.3 Keyword Selection 16

3.3.4 Example 19

3.3.5 Active Learning: Streaming 19

4 Results and Discussion 21

4.1 Results 21

4.2 Discussion 21

5 Conclusion 23

List of Figures

1 Relative frequency of various concepts in a corpus. Ranging from people, vehicles, weapons, specific landscapes, and specific buildings. 2
2 Overall system architecture 3
3 Example concept types considered by the system. 3
4 Shopforeman detected that the image was a photo taken outdoors in a natural setting. Since it is a photo it attempted to extract faces from the image but none were found. 4
5 Shopforeman detected that the image was a photo taken outdoors in a natural setting. Since it is a photo it extracted the face present. 5
6 Shopforeman detected that the image was not a photo (e.g., an image taken directly from a camera) and proceeded to detect if the image is a logo. It found that it is most likely an Apple logo using the Local Naive Bayes' Nearest Neighbor algorithm. 6
7 Directed Acyclic Graph (DAG) of a selection of the types used by the Shopforeman. 8
8 Indoor/Outdoor classifier performance on the SUN397 dataset. 9
9 (left) Proposed cascade training, query, and execution approach. (right) Example cascade stage instances. 12
10 (left) Recognition rates of features/kernels. (right) Existing (#0-3), upper bound (#4-5), and (#6-15) method variations (threshold method, max cascade length ℓ, cascade selection method). The threshold selection methods (Sec. 3.2.3) are $\mathcal{T}_e$ and $\mathcal{T}_{a,z}$. The cascade selection methods (Sec. 3.2.4) are D and S_α with β as half of the stages. 14
11 High level flow of the Nightwatchman's iterative process. 15
12 The interface provided to annotators for the label "outdoor". 17
13 The resulting annotations from Mechanical Turk for the label "outdoor". . . 18
14 High level overview of how data is stored in HBase for the Nightwatchman. . 19
15 Figure demonstrating the Nightwatchman flow for one iteration with results (i.e., pos/neg/chance/accuracy) included for the previously issued command sequence. 20

1. Summary

In this project we developed a prototype Shopforeman system that composes heterogeneous computer vision algorithms on the fly along with a complementary Nightwatchman system that automatically generates novel concept classifiers for the Shopforeman to use. The Shopforeman provides an interface to the user that allows them to specify the specific problem they are trying to solve and a target operating point. The goal of this project is to explore the design of an automated interface to a heterogeneous collection of computer vision algorithms and complementary approaches to develop concept classifiers and datasets that it uses.

We explore these issues broadly and present an approach that is capable of modeling complex computer vision systems while operating on large data collections using classifiers, image search indexes, human annotators, and heterogeneous computer vision algorithms. Processing is performed using the Apache Hadoop cluster execution framework and data is stored using Apache HBase. The role of this project is to explore an alternative approach to that taken by the TA1 and TA2 Visual Media Reasoning (VMR) participants and identify what components are most useful to the program overall. Our data storage (HBase), computer vision system (Picarus), and classifiers were transitioned to the TA2 teams.

2. Introduction

Many natural distributions follow a power law distribution and when viewed with entities sorted by frequency they form a long tailed distribution. For our purposes, the most frequent entities that occur in a database such as people, faces, and vehicles have substantial resources dedicated to detecting them accurately; however, outside of the few most common entities the rest are far less frequent and make up substantially more of the database's diversity. The challenge for our project is how to improve accuracy on the far end of this distribution where few samples exist for each entity and it becomes cost prohibitive to dedicate significant manual resources to developing custom algorithms for them. Figure 1 illustrates this concept graphically, where pictures of people occur frequently but pictures of a specific village may only occur rarely. In this project we develop algorithms that operate abstractly which allows our approach to be adapted to a wide range of visual classification tasks. To do this we automate standard experimental protocols to develop new concept classifiers and datasets using annotators to simplifying the interface for the operator.

Figure 2 provides an overview of the system architecture. The Shopforeman and Nightwatchman are the operator facing applications and use the open source Picarus machine learning system which abstracts data storage, processing, web crawling, and annotation tasks. The Shopforeman employs concept classifiers generated by the Nightwatchman. In turn, the Nightwatchman uses the available storage, processing, crawling, and annotation infrastructure to produce the concept classifiers.

The Shopforeman uses a type system to compose computer vision algorithms that satisfy the task input by the operator. For example, if we are given a series of photos and told to

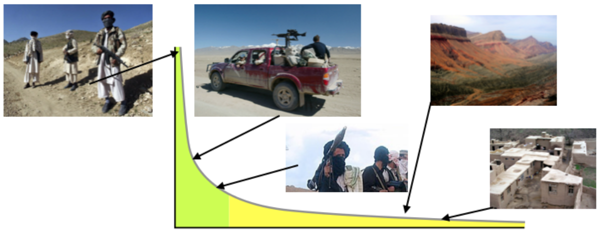


Figure 1: Relative frequency of various concepts in a corpus. Ranging from people, vehicles, weapons, specific landscapes, and specific buildings.

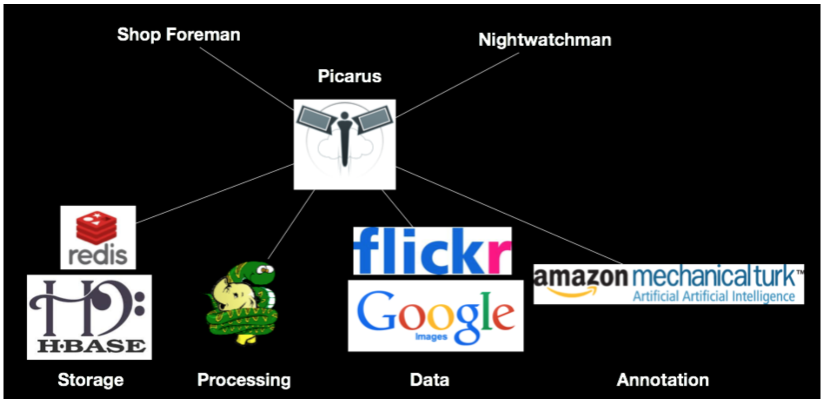


Figure 2: Overall system architecture

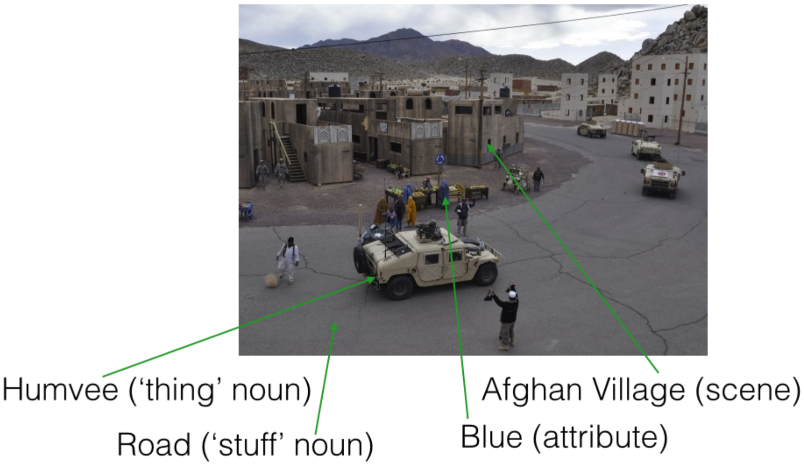
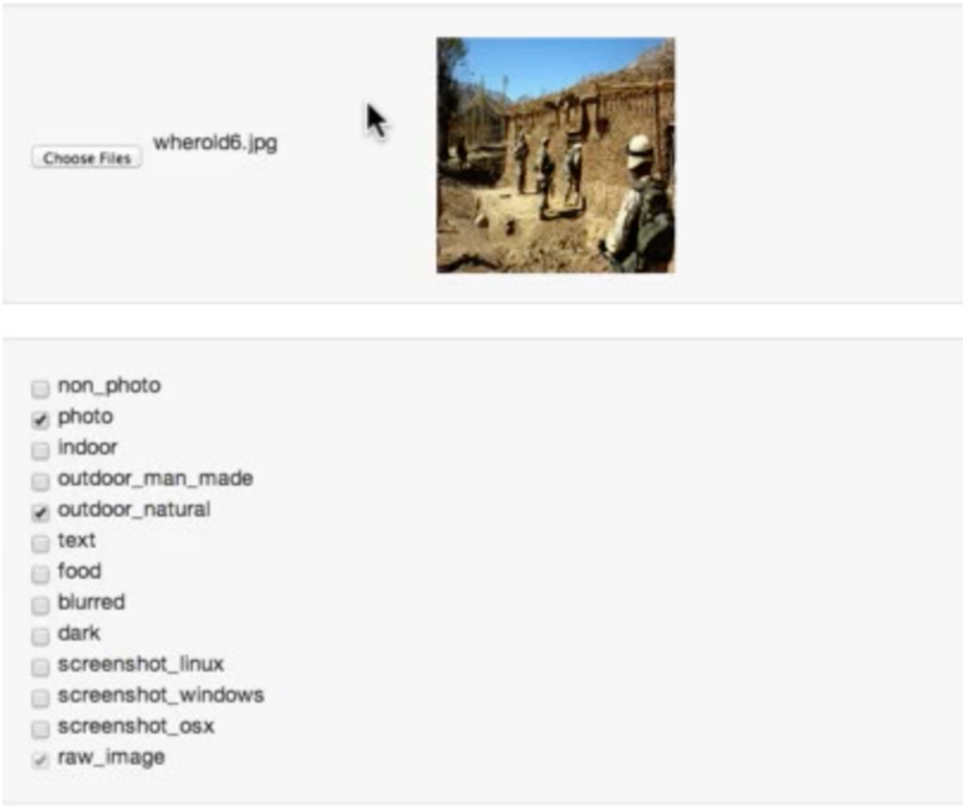


Figure 3: Example concept types considered by the system.

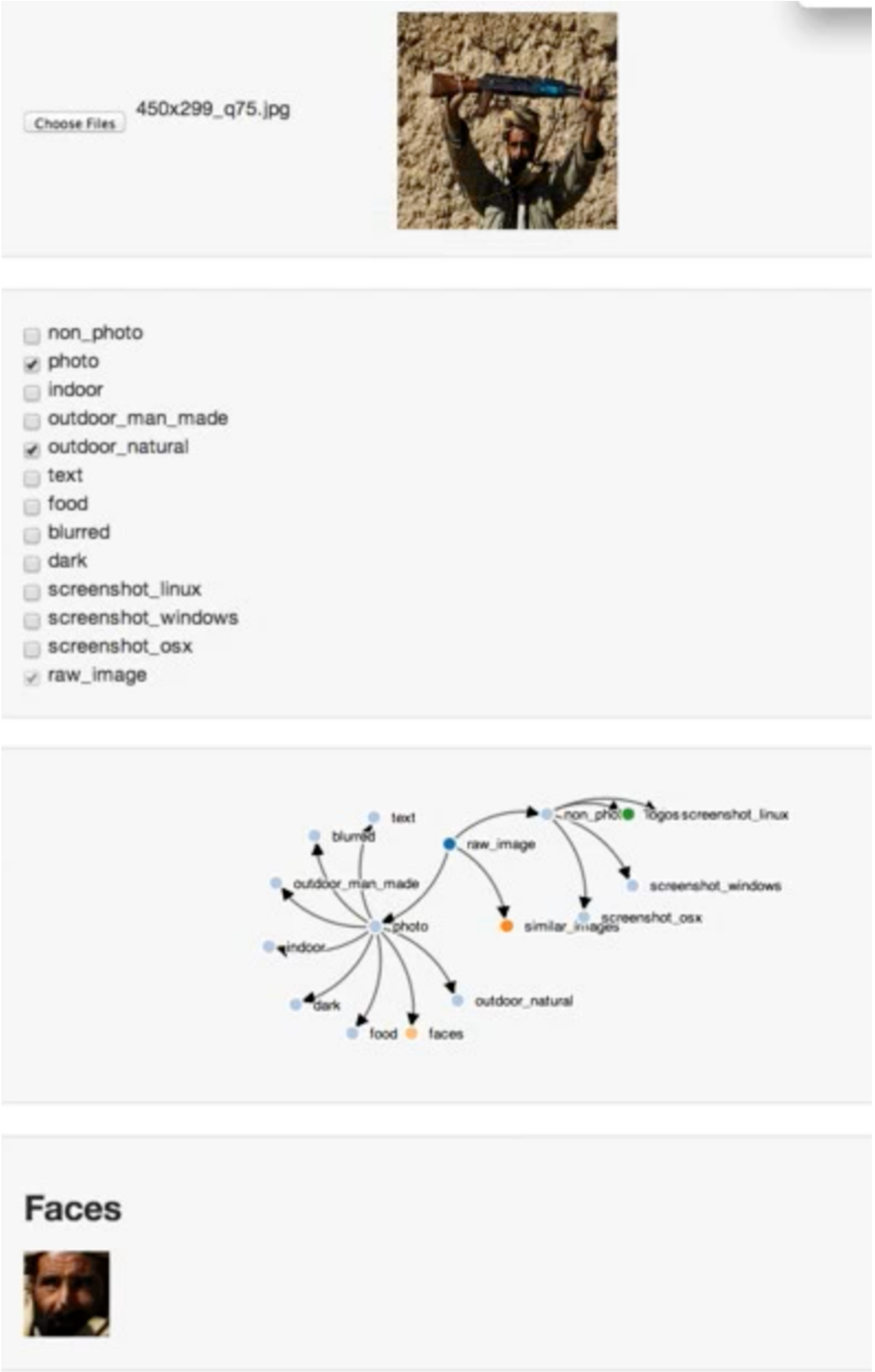
recognize the people that appear in them, the system knows that 1.) it must be a photo and not an abstract graphic (e.g., a drawing in a document), 2.) it must have a face in it, and 3.) after recognition we want a list of names of the people in the photo. This structure is highly flexible and allows for adding new algorithms and types. This structure abstracts the entire system into specifying input and output types, with the inference algorithm able to determine what approaches are available to satisfy it by performing a depth first search. The system is also able to refine the input type of an image for example by determining if it's indoors, a logo, or contains people. Figure 7 illustrates the different levels of concepts that we aim to classify.



wherold6.jpg

□ non_photo
☑ photo
□ indoor
□ outdoor_man_made
☑ outdoor_natural
□ text
□ food
□ blurred
□ dark
□ screenshot_linux
□ screenshot_windows
□ screenshot_osx
☑ raw_image

Figure 4: Shopforeman detected that the image was a photo taken outdoors in a natural setting. Since it is a photo it attempted to extract faces from the image but none were found.

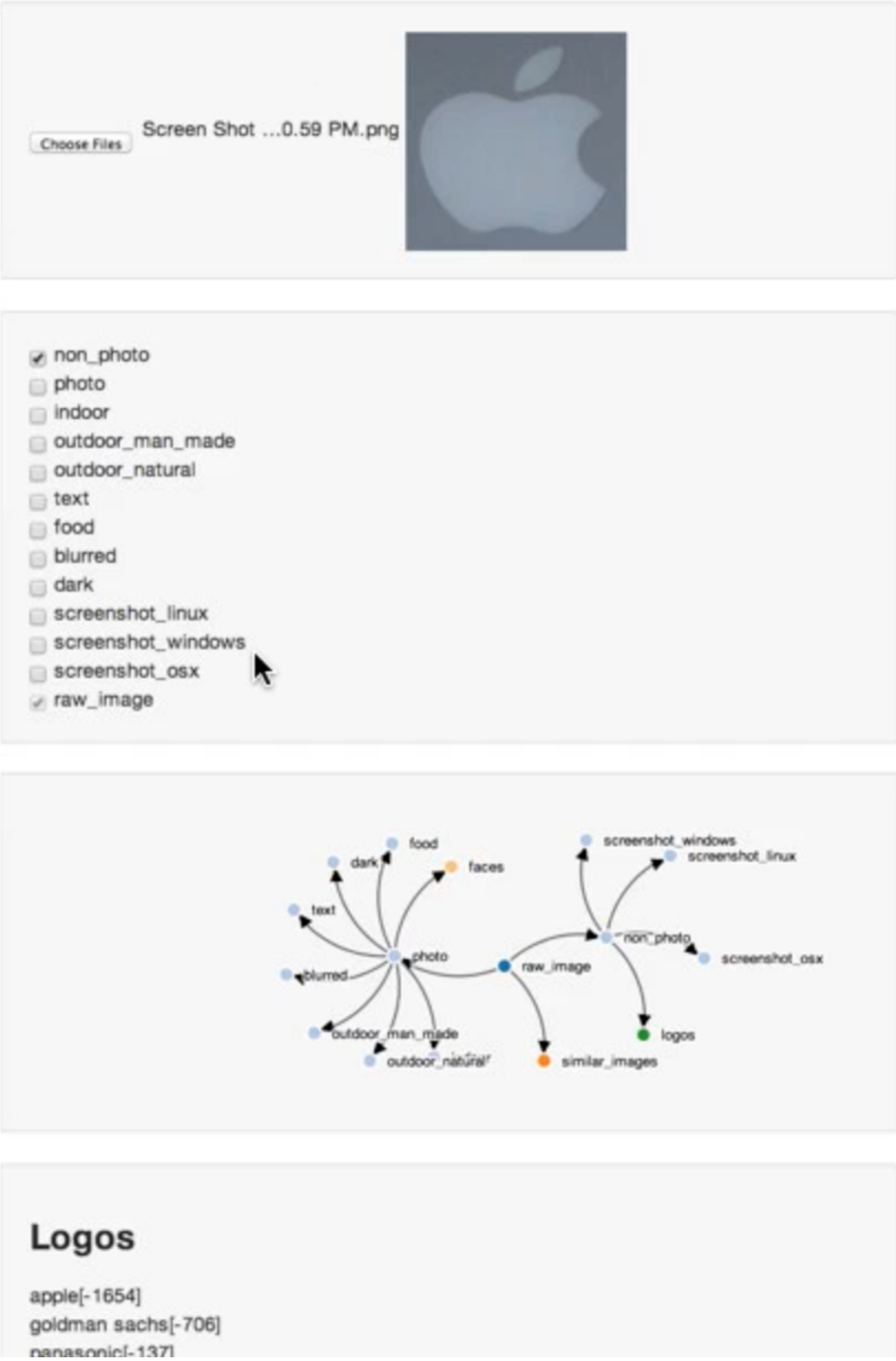


450x299_q75.jpg

☐ non_photo
☑ photo
☐ indoor
☐ outdoor_man_made
☑ outdoor_natural
☐ text
☐ food
☐ blurred
☐ dark
☐ screenshot_linux
☐ screenshot_windows
☐ screenshot_osx
☑ raw_image

Faces

Figure 5: Shopforeman detected that the image was a photo taken outdoors in a natural setting. Since it is a photo it extracted the face present.



☑ non_photo
□ photo
□ indoor
□ outdoor_man_made
□ outdoor_natural
□ text
□ food
□ blurred
□ dark
□ screenshot_linux
□ screenshot_windows
□ screenshot_osx
☑ raw_image

Logos

apple[-1654]
goldman sachs[-706]
panasonic[-137]

Figure 6: Shopforeman detected that the image was not a photo (e.g., an image taken directly from a camera) and proceeded to detect if the image is a logo. It found that it is most likely an Apple logo using the Local Naive Bayes' Nearest Neighbor algorithm.

3. Methods, Assumptions, and Procedures

3.1 Shop Foreman

3.1.1 Types

Each concept classifier has one input and output type. The type for newly input images is “raw image” and as concept classifiers positively identify additional attributes of the image it is further refined. The type system provides a straightforward way to determine if a task can be accomplished and what methods are available to do so. Moreover, the concept classifiers need to only operate over the domain of their input type (e.g., face recognition is only trained/evaluated on photos, not logos or screenshots) which improves their discriminative power substantially as compared to operating on possible images. This means that the type system implicitly induces a rejection chain cascade, where for an image to be evaluated by a classifier all preceding classifiers need to positively confirm their type. This configuration is widely used and is particularly useful at efficiently processing imagery as fewer classifiers are evaluated on each image.

3.1.2 Datasets

For machine learning tasks the data used for training and evaluation is often the key to achieving acceptable accuracy. Problems encountered include domain mismatch (e.g., training performed on a different domain that we evaluate with), training errors, intra-class variation (e.g., vehicles include busses, motorcycles, and cars), between class variation (e.g., bicycles and motorcycles are semantically different in day-to-day use but can look similar), and dearth of training samples. For our task the problem is more complex since long tail classes necessarily have few training samples and we are limited to crawling publicly available image repositories and datasets. While existing computer vision datasets are a good resource they tend to cover only common classes and are often heavily biased in ways that make extrapolating their performance in a real world setting challenging. However, collecting fully custom datasets for our task is impractical due to the wide scope and limited resources available. For our tasks we use a set of resources that support crawling by query (e.g., Flickr,

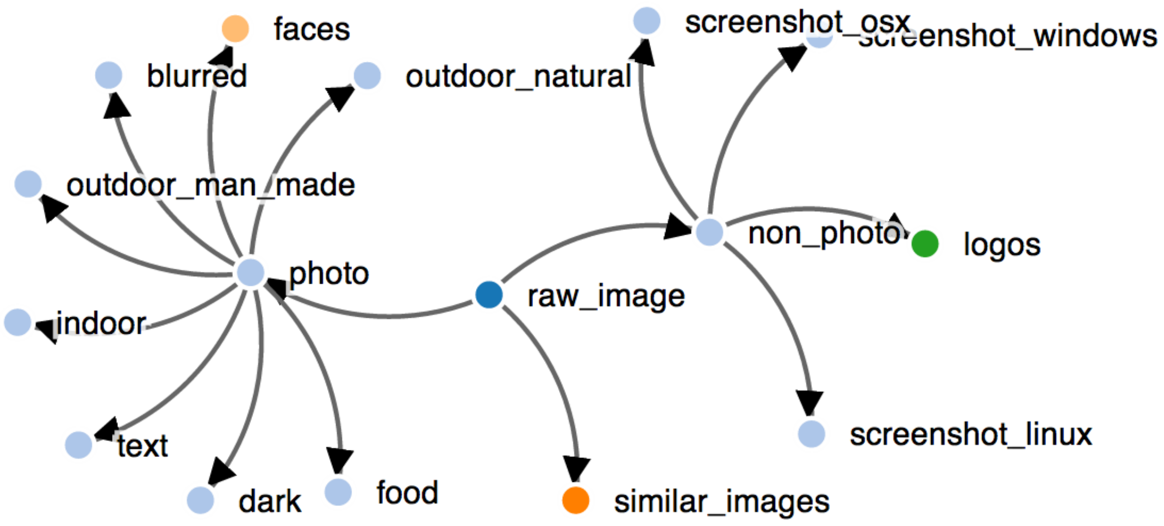


Figure 7: Directed Acyclic Graph (DAG) of a selection of the types used by the Shopforeman.

Google Maps, and Google Images) along with commonly used vision datasets (e.g., SUN397, Imagenet, and Labeled Faces in the Wild).

3.1.3 Components

While the concept classifiers are trained for novel classes, their underlying algorithms are drawn from a pool of generally applicable computer vision algorithms. For image classification features we use Gist[10], bag-of-visual-words, and spatial pyramid histograms[6]. For image classification we use support vector machines with histogram intersection and linear kernels. For image search tasks we use spherical hashing[4] to produce compact binary codes from feature vectors. For search indexing we use multi-index hashing[9] which enables sub-linear hamming distance computation.

We developed a new classifier based on the Local Naive Bayes Nearest Neighbor[8] method (a non-parametric nearest neighbor based classifier) for multi-class recognition problems such as landmark and logo recognition. This classifier suits this project well as it supports many classes with relatively few examples, compared to Support Vector Machines which have difficulty in supporting many classes and operating with few positive training examples.

Where: Indoor/Outdoor Scene Classification

As an illustrative example we show results on indoor/outdoor classification. The images used are from the SUN397 dataset with 19,850 images used for both training and testing. A linear SVM is used with bag-of-visual-words of histogram of oriented gradients features.

Curves detailing the Indoor/Outdoor classifier performance, with Indoor as the positive class and Outdoor as the negative class. The top 2 curves (unnormalized and normalized) are histograms of classification confidence, these show visually how separable the two classes are given the classifier's confidence value. The third curve shows the distribution of accuracy

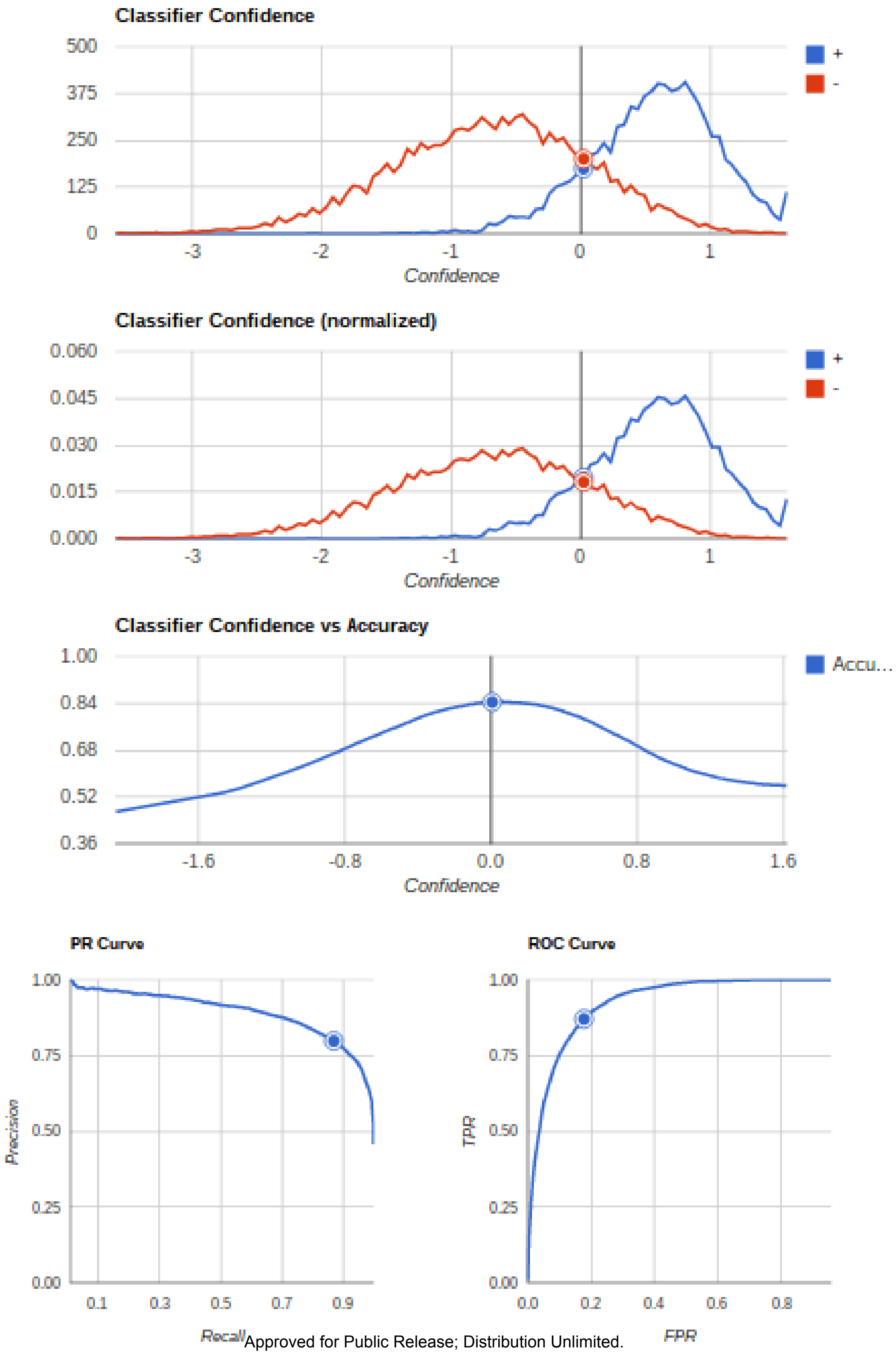


Figure 8: Indoor/Outdoor classifier performance on the SUN397 dataset.

(TP + TN) / (TP + TN + FP + FN) as a function of the confidence threshold. The SVM was set to treat errors equally. The circles in all of the curves corresponds to the same threshold (thresh = 0) which is where the accuracy is maximized. The last two figures are standard PR (TPR = Recall = (TP / (TP + FN)) and Precision = (TP / (TP + FP)) and ROC curves (TPR = Recall = (TP / (TP + FN) and FPR = (FP / (TN + FP)).

What/Where: Semantic Scene Parsing

For scenes that are primarily outdoors, knowing the basic structure of the scene (e.g., ground, sky, vegetation, man-made structure, water, etc.) can help significantly in determining high level scene properties (e.g., is there is a body of water, is there a building, etc.) and in determining scene classification labels (e.g., jungle, city, desert, ocean). To perform this task, we have an efficient scene parsing algorithm based on Semantic Texton Forests[12] that performs an efficient (sub-second) pixel level classification.

What: Logo Recognition

As an example of entity recognition we evaluated the Local Naive Bayes' nearest neighbor classifier on a dataset of the 100 top brands from Business Week (query Google Images with human verification) with 486 (80%) of the images used for training and 124 images (20%) used for evaluation. The image features used are multi-scale LAB color space image blocks. Random chance on this problem is approximately 1% and this configuration achieved 49% accuracy (i.e., the first ranked result is correct).

What: Landmark Recogntiion

Another example of entity recognition is demonstrated on recognizing one of four DC landmarks (lincoln memorial, thomas jefferson memorial, us capitol, washington monument) from Flickr images. The training set consisted of 232 (80%) images and the testing set had 54 (20%) images. Chance in this task is approximately 25% and we evaluated this approach on a variety of image features: Dense HOG (16x16) 70%, Dense HOG (32x32) 79%, Dense HOG (64x64) 79%, SURF 83%. In this task the images are unconstrained and have similar landscapes as they are taken from the same region, no task specific tuning was done so that we can evaluate the baseline classification performance which can be expected from the concepts automatically generated by the Nightwatchman.

3.1.4 Picarus

The underlying requirements of having algorithmic and human modules for the Shop Foreman to reason over necessitates an early focus on developing base implementations of these modules. The goal is to produce a sufficient number of modules to demonstrate the effectiveness of our Shop Foreman algorithm. To facilitate this, we are using the open-source Picarus library (Brandyn White is the primary developer) to produce classifiers and search indexes that are the core of the underlying algorithmic modules and to facilitate interactive human annotation for human input (both by users and by annotation workers). Picarus is a web-service that executes large-scale visual analysis jobs using Hadoop with data stored on

HBase. This approach simplifies integration with TA1/TA2 teams as they can be given access to the web-service directly which has a REST (common HTTP data-access convention) interface and web application.

As the entire backend is available open source and we installed a Picarus cluster on Amazon's Gov Cloud for participating teams to use. Adapted existing security, administration, and data management to VMR requirements. Installed Picarus (which requires Hadoop, HBase, and Redis) on two govcloud servers. Wrote up documentation for picarus administration http://goo.gl/fDWqhV. Frequent interactions and consultation with SET/Blackriver teams to transfer control of the server and ensure proper instructions were available for expected tasks.

3.2 Cascade Calibration

We introduce a classifier and parameter selection algorithm for Classifier-as-a-Service applications (i.e., the model of our Shopforeman and Nightwatchman) where there are many components (e.g., features, kernels, classifiers) available to construct classification algorithms. Queries specify varying requirements (i.e., quality and execution time), some of which may require combining classification algorithms to satisfy; each query may have a different set of quality and execution time requirements (e.g., fast and precise, slow and thorough) and the set of images to which the classifier is to be applied may be small (e.g., even a single image), necessitating a query resolution method that takes negligible time in comparison. When operating on large datasets, meeting design requirements automatically becomes essential to reducing costs associated with unnecessary computation and expert assistance. As queries specify requirements and not implementation details, additional components can be utilized naturally. Our query resolution method combines classifiers with complementary operating points (e.g., high recall algorithmic filter, followed by high precision human verification) in a rejection-chain configuration. Experiments are conducted on the SUN397[14] dataset; we achieve state-of-the-art classification results and 1 m.s. query resolution times.

This cascade calibration approach is used throughout our system. The Shopforeman employs it to resolve which concept classifier to use for a given task. The Nightwatchman uses it for selecting which concept classifiers formed in a cascade are most effective at satisfying an operator provided target operating criteria.

3.2.1 Proposed Approach

We describe a method for efficiently determining a combination of classifiers and thresholds sufficient to achieve a desired per-class quality and overall execution time specification, provided as a "query". Query resolution involves combing classifiers into a rejection-chain cascade which satisfy quality and execution time requirements as close as possible. Our primary contribution is the development of an efficient method to determine the performance characteristics of these cascades offline, so that they can be queried efficiently online. Given pre-trained cascade stages $\mathcal{S}$ (Sec. 3.2.2), we predict each class $l \in \mathcal{L}$ using each cascade stage $S \in \mathcal{S}$ producing a confidence matrix $\vec{x}^S$ with $|\mathcal{L}|$ rows and N columns, where N is the size of a validation set. We illustrate our approach on the scene recognition task, where each image belongs to exactly one class and the validation groundtruth $\vec{g}$ is a matrix of the same

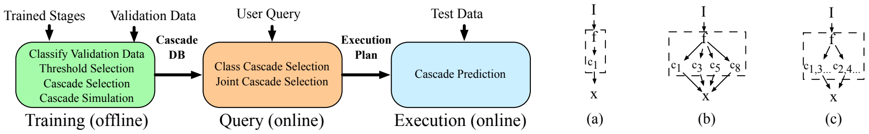


Figure 9: (left) Proposed cascade training, query, and execution approach. (right) Example cascade stage instances.

shape as $\vec{x}^S$ with values $\vec{g}_{l,i} \in \{-1, 1\}$ for validation image i. For each cascade stage S the threshold selection (Sec. 3.2.3) produces a set of thresholds $\mathcal{T}_l^S$ that capture that cascade stage's performance. The cascade selection produces $\mathcal{C}_l \subseteq \mathcal{P}(\mathcal{S})$ where $\mathcal{P}(\mathcal{S})$ is the powerset of the set of cascade stages $\mathcal{S}$. Finally the cascade simulation (Sec. 3.2.4) *simulates* each cascade $C \in \mathcal{C}_l$ using the thresholds selected $\mathcal{T}_l^S$ where $S \in C$. The simulation is an efficient method of evaluating the performance of the cascades and produces operating points (i.e., confusion matrices and times) that are then stored in the *cascade database*. This training phase operates on each class l independently and, for notational convenience, we let $\vec{g} = \vec{g}_l$, $\vec{x}^S = \vec{x}_l^S$, $\mathcal{T}^S = \mathcal{T}_l^S$, and $\mathcal{C} = \mathcal{C}_l$.

3.2.2 Cascade Stages

A cascade stage S takes an image I and a set of classes $\mathcal{L}$ from which it produces a (possibly sparse) confidence vector x^S. For the single class case in Fig. 9(a), this is a single image feature f and a single binary classifier c_1 which produces the confidence vector x^S as $x_1^S \leftarrow c_1(f(I))$. For the general form in Fig. 9(b), a single feature f is shared by many classifiers and x^S is produced as $\forall l \in \mathcal{L}, x_l^S \leftarrow c_l(f(I))$. We now provide a definition for a cascade stage S where I is an image and $\mathcal{L}$ is the set of classes (represented as positive integers): (1) The cascade stage is defined for all $\mathcal{L}$, $x^S \leftarrow S(I, \mathcal{L})$ where S may be non-deterministic and x^S is a real-valued vector, (3) $\forall l, m \in \mathcal{L}$, confidence value x_l^S is independent of $m \in \mathcal{L}$ or $m \notin \mathcal{L}$ where $l \neq m$, and (4) Larger values of x_l^S signify higher confidence that the input belongs to class l.

3.2.3 Threshold Selection

Given a trained cascade stage S and a validation set, our task is to find a set of thresholds $\mathcal{T}^S$ that compactly represents its operating points. The threshold selection occurs independently for each class and cascade stage S. We wish to minimize the number of thresholds $|\mathcal{T}^S|$ to reduce the cascade simulation complexity. We say a confidence value $\vec{x}_i^S$ is positive with respect to $t \in \mathcal{T}^S$ if $\vec{x}_i^S \geq t$. The initial set of thresholds is $\mathcal{T}_u^S = \mathcal{X}^S \cup \{\infty\}$, where $\mathcal{X}^S$ is the set of confidence values of $\vec{x}_S$. Including infinity ensures that at least one threshold has no false positives. This set is *sufficient* as any other threshold produces a redundant partition; however, it is not *necessary* as thresholds which produce "worse" confusion matrices may be present (i.e., more FP or FN errors). We construct a subset of $\mathcal{T}_u^S$ that is both necessary and sufficient. Given a confidence value $\vec{x}_i^S$ and ground truth label $\vec{g}_i$ for each image i in the validation set, we sort them ascending by confidence value with positive ground truth instances listed before negative ones for the same confidence values. Observe that the secondary sorting eliminates ‘overestimated’ confusion matrices that can result from naïve generation of confusion matrices with the same confidence value but different ground truth polarities[1]. As this process is independent of the stage S, we let $\bar{\vec{g}} = \bar{\vec{g}}^S$, $\bar{\vec{x}} = \bar{\vec{x}}^S$, and $\mathcal{X} = \mathcal{X}^S$. When operating on the vector $\bar{\vec{g}}$, $\bar{\vec{g}}_i$ represents the ground truth polarity at position i, with $\bar{\vec{x}}_i$ as its associated confidence value and $\bar{\vec{g}}_{i-1}$ as its neighbor in the descending direction.

We find the minimum number of thresholds $\mathcal{T}_e$ required to exactly represent the performance characteristics of the cascade stage. The predicate $Keep(\bar{\vec{x}}_i)$ is true when $\bar{\vec{g}}_{i-1}$ is not positive and $\bar{\vec{g}}_i$ is not negative; this clearly produces the desired set as $\mathcal{T}_e = \{\bar{\vec{x}}_i \in \mathcal{X} : Keep(\bar{\vec{x}}_i)\}$. More practically, we can relax our exact representation by allowing for a bounded absolute precision/recall difference, z, between the confusion matrices produced by $\mathcal{T}_e$ and a subset $\mathcal{T}_{a,z}$ of them, leading to $O(1)$ thresholds. This is accomplished by assigning each threshold in $\mathcal{T}_e$ as a node in a graph with edges induced by z and computing the dominating set, which correspond to thresholds in $\mathcal{T}_{a,z}$.

3.2.4 Cascade Selection/Simulation

We evaluate two methods for selecting which candidate cascades of length ℓ should be used for simulation: dense (D) and sparse (S). The sparse method selects the β stages with highest rank correlation to the ground truth and limits the branching factor to α stages with lowest minimum rank correlation with preceding stages. Given a set of cascades $\mathcal{C}$ for a class with an associated set of thresholds $\mathcal{T}^S$ where $S \in C$ and $C \in \mathcal{C}$, the goal is to efficiently compute a *cascade database* that contains the union of all cascades in $\mathcal{C}$ that can be formed over all of their thresholds. This process is a simulation as we are not computing the cascade performance directly; rather, we find the confidence values for all stages independently and exactly compute their combined performance offline. The output for each set of thresholds for a cascade includes a binary confusion matrix, stage names, stage thresholds, and % of inputs evaluated at each stage (used to compute time).

A naïve approach would evaluate the validation set using every possible chain and its corresponding set of thresholds. There are a combinatorial number of operations in the length of the cascade if every possible ordering of a single cascade (e.g., $A \rightarrow B \rightarrow C$, $C \rightarrow B \rightarrow A$) were considered; however, the order does not effect the resulting binary confusion matrix. We order the stages by running time per image. When evaluation a cascade $A \rightarrow B \rightarrow C$, we have already computed A and $A \rightarrow B$ which can be re-used for longer cascades with common prefixes (e.g., $A \rightarrow B$ reused in $A \rightarrow B \rightarrow C$ and $A \rightarrow B \rightarrow D$). Stages with the same parent are all computed simultaneously. The computational complexity of this method is the sum of the complexity for each cascade from a leaf to a root node. The complexity of each path is bounded by $O(N \prod_{S \in C} |\mathcal{T}^S|)$ where N is the number of validation inputs (e.g., images for image classification). However, this worst case performance cannot occur since the number of validation inputs N reduces from stage to stage, exactly as they do when passing through a rejection chain cascade.

		Rec. Rate	
Name	Dim.	Lin.	HIK
$\sum_F$ K[3]	-	22.5	29.0
OB[7]	44K	18.7	25.1
HOG2x2[14]	6321	17.2	26.5
GIST[11]	960	10.5	12.7
LABhist	41K	3.6	10.4
Texton[12]	1785	3.1	6.5
AutoC.[5]	256	4.8	5.9
Tiny[13]	3072	2.1	5.0

		Mean	# AUC [14]			Sim.	Mean #
#	Method	AUC	>	=	<	(sec.)	$\|\mathcal{T}\|$,$\|\mathcal{C}\|$,$\|DB\|$
0	SUN397[14]	.957	0	397	0	-	-
1	GIST[11]	.843	1	0	396	-	-
2	OB[7]	.923	15	26	356	-	-
3	$\sum_F$ K[3]	.934	23	28	346	-	-
4	U.B. $\ell = 1$	.954	91	117	189	0.021	40,16,639
5	U.B. $\ell = 2$	.969	216	138	43	2.148	40,136,117K
6	$\mathcal{T}_e$,1,D	.949	59	94	244	0.023	41,16,651
7	$\mathcal{T}_e$,2,D	**.963**	**157**	130	110	2.439	41,136,123K
8	$\mathcal{T}_e$,2,S_2	.956	105	112	180	0.157	41,19,8135
9	$\mathcal{T}_{a,.05}$,1,D	.938	25	40	332	0.010	13,16,201
10	$\mathcal{T}_{a,.05}$,2,D	.949	64	78	255	0.235	13,136,12K
11	$\mathcal{T}_{a,.05}$,2,S_2	.942	34	58	305	0.024	13,18,1095
12	$\mathcal{T}_{a,.05}$,3,S_2	.944	42	63	292	0.156	13,28,6296
13	$\mathcal{T}_e$,2,D	.965	175	135	87	2.629	41,153,146K
14	$\mathcal{T}_{a,.05}$,2,D	.950	69	87	241	0.276	13,153,14K
15	$\mathcal{T}_{a,.1}$,5,S_1	.925	5	19	373	0.103	8,21,1761

Figure 10: (left) Cascade recognition rates of features/kernels for our approach. (right) Existing (#0-3), upper bound (#4-5), and (#6-15) method variations (threshold method, max cascade length ℓ, cascade selection method). The threshold selection methods (Sec. 3.2.3) are $\mathcal{T}_e$ and $\mathcal{T}_{a,z}$. The cascade selection methods (Sec. 3.2.4) are D and S_α with β as half of the stages.

3.2.5 Experiments

We show experimental results on the SUN397[14] scene recognition dataset which consists of 39,700 images (50 train/test per class, using partition #1). Fig. 10(a) summarizes the features used along with their recognition rate for multi-class classification using linear and histogram intersection kernels (HIK) over all of the features. LABhist uses the CIE L*a*b* colorspace with 4 bins for L and 11 bins for a and b. TextonForest uses the method proposed in [12] (trained on the MSRC dataset) to compute spatial pyramids on the maximum label mask. The TinyImage[13] feature is a common baseline for scene recognition. The image is resized to 32x32 in the CIE L*a*b* colorspace. ObjectBank (OB)[7] is a method that pools object detector predictions to produce a highly discriminative feature. Spatial pyramids (used by HOG2x2, LABhist, and Texton) are of scales $2^{0 \cdots 3}$. Using a spatial pyramid significantly improves performance when used with HIK at marginal additional computational cost. Mechanical Turk indoor classification is included as a cascade stage with confidence estimated from response time Fig. 10(#13-15), the best algorithmic result is bold.

As we are evaluating binary classification performance, we use the mean area under the ROC. Our method produces a significant improvement (compared to Fig. 10(#0-3)) using similar features, classifiers, and kernels Fig. 10(#7). It is clear that this gain is from the cascade design as the features alone Fig. 10(#6) perform worse. We use 20% of the training set as a validation set to learn the *cascade database* with the classifiers trained on the other 80%. We are able to calculate an upper bound for our method by identifying the optimal cascades and thresholds on the test set Fig. 10(#4-5); the approach has near ideal performance. To gain further insight we show the number of classes with AUC better, same, or worse (>, =, <) than [14] for each result in Fig. 10 (our method uses 20% less training than [14] due to calibration).

To evaluate query performance, for all classes and for constraints of length 1 to 5 (i.e., precision, recall, time, accuracy, F-1) we compute 100 random example queries (e.g., House with $p > .4$, $r > .3$, and $t < 1$) with a cap of 100 returned cascades using the *cascade* *database* produced by Fig. 10(#14). The median query times (in ascending order from 1-5 constraints) are: 1.1, 3.5, 1.0, .1, and .09 in m.s.; The additional constraints reduce the search space dramatically. This is fast compared to the feature/classifier computation and is only performed once per query resolution, not per image as in[2]. Multi-class queries (i.e., quality per class, overall time) produce per-class candidates and the overall time is minimized. Computations were performed on a 2.2GHz Xeon.

3.3 Nightwatchman

The Nightwatchman produces concept classifiers for the Shopforeman to use. To do this it orchestrates an iterative crawl, annotate, train, and evaluate loop. Figure 11 shows the high level steps that the Nightwatchman performs. The initial phase updates which slices of data should be annotated for each of the training/validation sets. The updated slices are annotated and classifiers are created from the annotated training samples. The trained concept classifiers are then used to predict the class labels of the validation set and from these prediction values we are able to calibrate the classifiers, using the previously described approach, which allows us to know how effective they are and what thresholds are required to achieve our target operating point.

3.3.1 Annotation

Many of the High-Level tasks (i.e., those that inform a Who/What/Where/When such as landmark identification) require significant training data, and few existing databases have relevant imagery and fewer have relevant annotation. Our approach has been to utilize large imagery collections such as Flickr and Google Images to obtain an initial dataset and then manually annotate them using Mechanical Turk.

Besides simplifying the data collection process, by observing how the annotators interpret their prompts and directions, we are able to better understand the task we are attempting to automate. For example, when asked to annotate furniture on a diverse dataset, annotators often disagreed on what that meant. Chairs, tables, and beds were obvious, but what about pool tables, park benches, kitchen counters, and arcade machines? This is a sign that furniture may not be a suitable class and perhaps finer grained classes (bed, chair, table, etc.) would not only simplify annotation but also simplify interpretation of the classifier's results in real-world use.

Figure 12 shows the interface provided to annotators. It includes a high level description of the annotation process and a class specific description that disambiguates any complex cases that may cause annotators to disagree on what label an image should be. This interface is for binary class labels to allow for quickly annotating a large set of images, requiring only a single button press per annotation with new images preloaded to instantly produce a new unlabeled image. Figure 13 shows the resulting annotations for "outdoor".

3.3.2 Active Learning: Batch

In batch active learning mode we evaluate a corpus for the samples that give us the most information if they are annotated. Three distinct sets of data are collected for annotation:

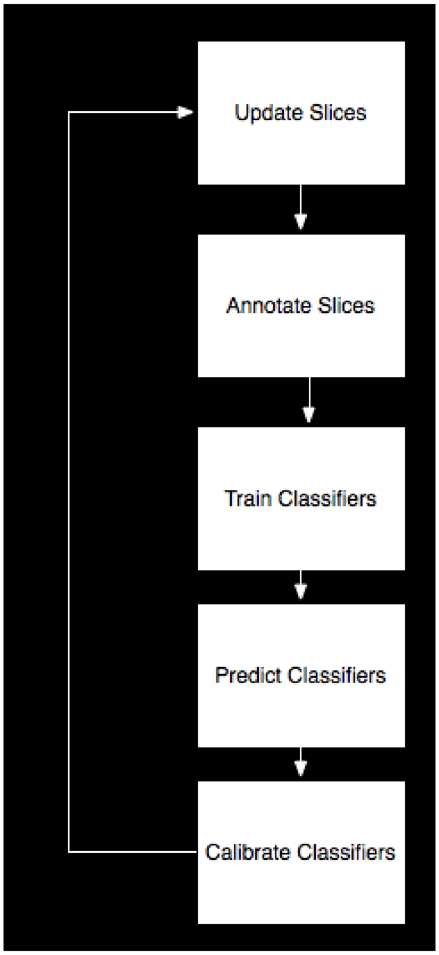


Figure 11: High level flow of the Nightwatchman's iterative process.

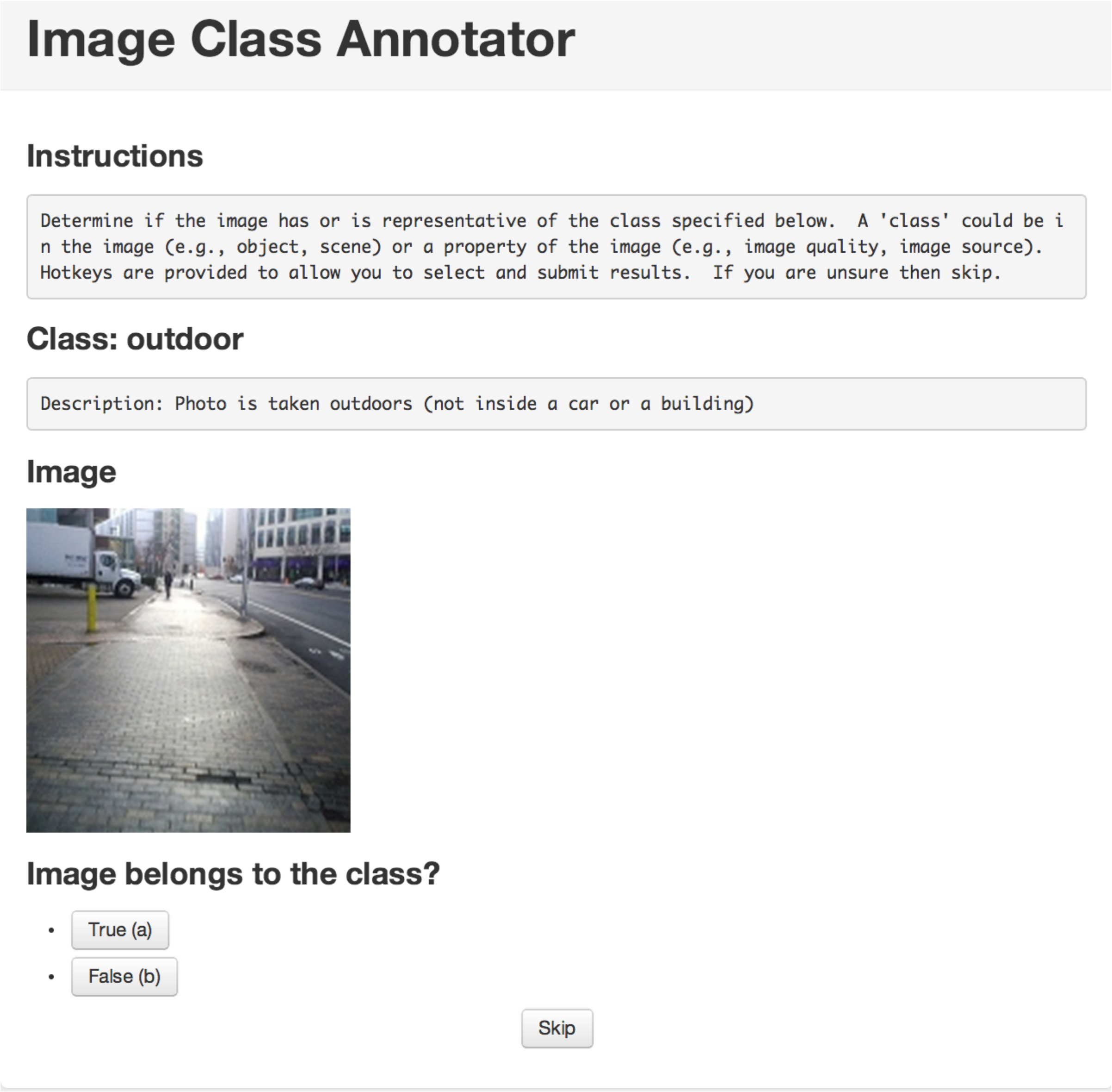


Image Class Annotator

Instructions

Determine if the image has or is representative of the class specified below. A 'class' could be in the image (e.g., object, scene) or a property of the image (e.g., image quality, image source). Hotkeys are provided to allow you to select and submit results. If you are unsure then skip.

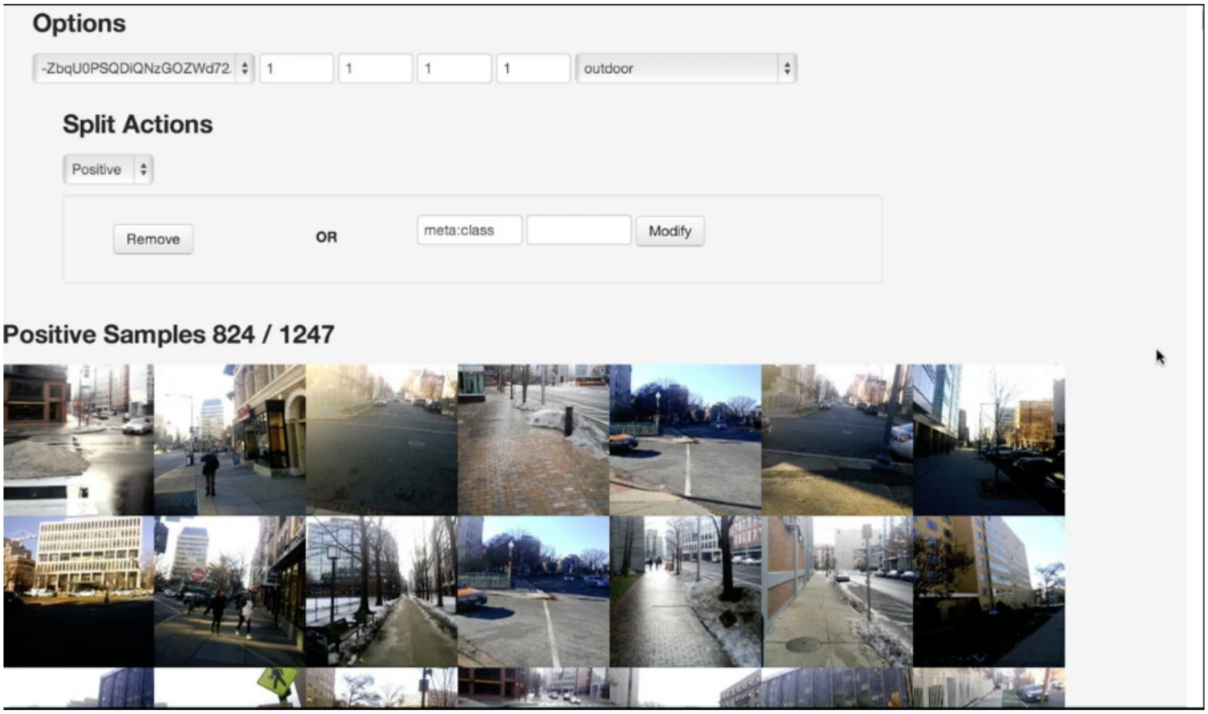
Class: outdoor

Description: Photo is taken outdoors (not inside a car or a building)

Image

Image belongs to the class?

Figure 12: The interface provided to annotators for the label "outdoor".



Options

-ZbqU0PSQDiQNzGOZWd72. 1 1 1 1 outdoor

Split Actions

Positive

Positive Samples 824 / 1247

Figure 13: The resulting annotations from Mechanical Turk for the label "outdoor".

uniformly random sample of validation images, uniformly random sample of training images, and a sample of strongly predicted positives/negatives. This strategy ensures that 1.) the validation set is randomly sampled which allows us to evaluate performance statistics independent of the concept classifiers effectiveness, 2.) the training is likely to have a balanced set of positive/negative examples, and 3.) if the concept classifiers perform poorly the uniform sampling will overcome any systemic biases in the long run. The features and classifiers are those previously specified.

3.3.3 Keyword Selection

When given a large set of input classes they will, in general, not all be effective candidates for automatic discovery. For example, it is common for photo tags to have entirely non-visual entries such as the name of the photographer. Ideally we could prune these to make more efficient use of our available resources. Another failure mode is when the image features extracted don't capture the particular defining characteristics of the class, such as a picture being "exciting" which may have primarily visual characteristics but is unlikely to be captured with low level image features. Semantic ambiguity (e.g., is a pool table a table?) can cause annotators to disagree about what class an image is and the resulting classifier will at least as unsure. The datasets being mined may not have any positive or even negative instances (e.g., if the dataset is all outdoors and the class is 'outdoors') of the class. As the learning process is continuous and iterative, we will eventually reach the useful limit for a given class and should focus on other classes. These failure modes and others are avoided by tuning the selection algorithm to focus on classes with 1.) high inter-annotator agreement,

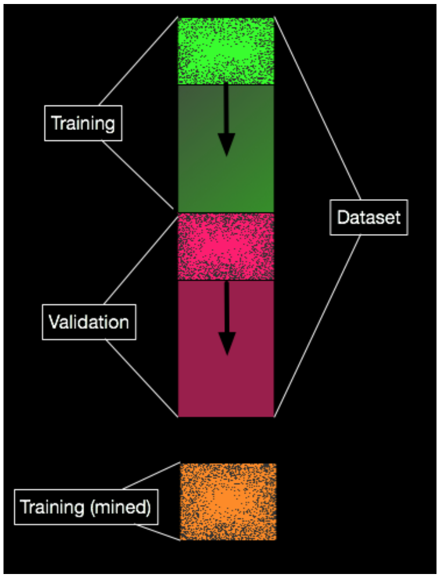


Figure 14: High level overview of how data is stored in HBase for the Nightwatchman.

2.) high annotator and classifier agreement, and 3.) have a significant marginal gain in accuracy between iterations.

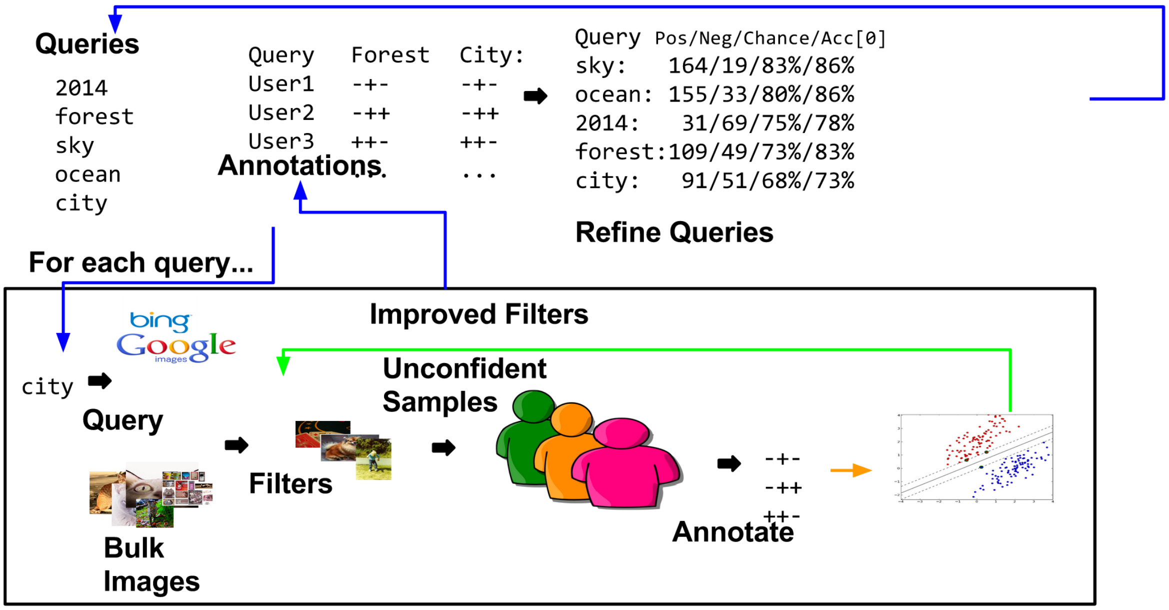
This is closely related to a classical problem called multi-armed bandit problem, where there are a row of slot machines with unknown payout rates and you have a fixed number of attempts to use them with the goal of maximizing your total reward. This results in a tradeoff between exploration (e.g., attempting to estimate the payout rate of as many machines as we can) and exploitation (e.g., use the machine that has the highest known payout rate). In our situation we can devise an objective function that models our overall satisfaction with the results (e.g., # of classes that meet specified target operating points). This approach is able to dynamically add classes and datasets during operation.

Figure 14 shows how data is annotated and evaluated by the Nightwatchman. Images are stored in HBase which allows them to be store contiguously on disk for high throughput processing. The initial dataset is partitioned into training and validation slices and we annotate images in descending order from the beginning of the key space. A separate contiguous training set that corresponds to hard samples mined from a large corpus is maintained to further improve classification.

3.3.4 Example

Next we provide an example of how classes are added to the Nightwatchman and the annotation, classification, and evaluation loop is initiated. The bulk images are crawled from Flickr and five popular tags are used: 2014, City, Forest, Ocean, and Sky.

```
python nwm.py add_task bwhitenwm: bwhite@cs.umd.edu 5
python nwm.py add_class zvjsox8ZTQu2tNUMJs-kkA sky "" --query sky
python nwm.py add_class zvjsox8ZTQu2tNUMJs-kkA 2014 "" --query 2014
python nwm.py add_class zvjsox8ZTQu2tNUMJs-kkA city "" --query city
```



```
Query Pos/Neg/Chance/Acc[0]
sky:    164/19/83%/86%
ocean: 155/33/80%/86%
2014:    31/69/75%/78%
forest:109/49/73%/83%
city:    91/51/68%/73%
```

Figure 15: Figure demonstrating the Nightwatchman flow for one iteration with results (i.e., pos/neg/chance/accuracy) included for the previously issued command sequence.

```
python nwm.py add_class zvjsox8ZTQu2tNUMJs-kkA ocean "" --query ocean
python nwm.py add_class zvjsox8ZTQu2tNUMJs-kkA forest "" --query forest
python nwm.py batch bwhite@cs.umd.edu zvjsox8ZTQu2tNUMJs
```

3.3.5 Active Learning: Streaming

The previous active learning approach operates on a large corpus of existing data and samples are selected for annotation in batch; however, we can pose the problem differently and instead of deciding what samples to present to annotators we can allow them to explore the corpus at will and determine what questions to ask based on what they are currently viewing. This approach allows us to continuously collect annotations as analysts are browsing the dataset. In this way we can leverage their existing workflow and they are likely to have a deeper understanding about the imagery they choose to view instead of uniform selection. Moreover, this approach reduces their cognitive load by having questions available while they are exploring the data and it naturally improves performance for data that is frequently accessed.

4. Results and Discussion

4.1 Results

The results for each component are provided in their individual sections and summarized here. In Section 3.2.5 we show that by composing independently trained binary classifiers in a classifier cascade we can achieve state-of-the-art performance on the scene recognition task. Our algorithmic components, described in Section 3.1.3, demonstrate that using our computer vision architecture (Picarus) we are able to build binary/multiple class classifiers and search indexes without writing task-specific code. Our logo recognizer we achieved 49% accuracy (i.e., the first ranked result is correct) with random chance on this problem as 1%. Our landmark recognizer achieved 83% accuracy on photos taken 'in the wild' of DC landmarks with chance being 25%. Our indoor/outdoor classifier achieved an accuracy of 85% with change being 50% on the SUN397 dataset.

4.2 Discussion

Throughout this project we learned several lessons that would likely be applicable to future applications. This section summarizes lessons learned that are more cross cutting in nature and weren't explored in other sections.

When dealing with large-scale open-ended problems such as the one considered in this work, it is important to impose constraints that prevent undue effort from being spent on the most difficult classes. Often the relative effectiveness of a classifier can very greatly between classes and it is much more efficient to start from the easiest classes and move to more difficult ones when their performance saturates.

Human annotation is expensive and it is worth optimizing the annotator's workflow to reduce cognitive load and judgement calls. Complex definitions make it substantially more difficult to annotate, especially for short term annotators. It is much more effective to provide an intuitive and judgement-free definition which may require breaking one complex class into two more clearly distinguished classes (e.g., vehicle vs 4-wheeled vehicle) or combining classes (e.g., car vs toyota).

Classifiers aren't perfect and the simpler the domain they operate in the more effective they will be, this is especially true when only a small training set is available. For example, if every classifier has to properly handle non-photos (e.g., logos, heavily edited photos, screenshots, documents) then it must necessarily be more complex; however, if instead we restrict their domain to photos they will generally be more accurate. Moreover, by reducing the domain size through the use of rejection chain cascades, as we do implicitly with our Shopforeman's type system, even weak classifiers can jointly form a strong classifier.

The intense research effort spent on individual common classes (e.g., person, car, face) is intractable for the long-tail of visual concepts. Instead of focusing on a single class's accuracy it's more effective to look at how many classes achieve an operationally useful classification accuracy. By focusing on what our methods are able to classify we can then use statistical models to infer attributes about more relevant but not as easily observed classes (e.g., using scene classifiers to infer the types of objects that it may contain, such as a bathroom having a sink).

5. Conclusion

In this project we developed a prototype Shopforeman system that composes heterogeneous computer vision algorithms on the fly. The Shopforeman provides an interface to the user that allows them to specify the specific problem they are trying to solve and a target operating point. Additionally, we developed a complementary Nightwatchman system that automatically generates novel concept classifiers for the Shopforeman to use by iteratively mining available data resources (e.g., Flickr), selecting the most informative samples to annotate, hire remote annotators using mechanical turk, and then train classifiers based on the annotation results.

Throughout our period of performance we worked with TA1 and TA2 participants to integrate components of our approach including our classifiers, data storage system (Apache HBase), and our in-house computer vision system (Picarus). We identified a need for a way to automatically populate the VMR system with additional concept classifiers as manually creating them is impractical for the "long tail" of concepts. Moreover, we demonstrated a scalable approach to collecting a dataset for each concept while simultaneously learning classifiers for it using only computational resources and untrained human annotators.

Bibliography

[1] T. Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[2] T. Gao and D. Koller. Active classification based on value of classifier. In *NIPS*, 2011.

[3] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *ICCV*, pages 221–228, 2009.

[4] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. Spherical hashing. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2957–2964. IEEE, 2012.

[5] Huang, Jing, Kumar, S. Ravi, Mitra, Mandar, Zhu, Wei J., and Zabih, Ramin. Image Indexing Using Color Correlograms. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 762+, Washington, DC, USA, 1997. IEEE Computer Society.

[6] Lazebnik, S., Schmid, C., and Ponce, J. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2169–2178, Los Alamitos, CA, USA, October 2006. IEEE Computer Society.

[7] L. Li, H. Su, E. Xing, and L. Fei-Fei. Object bank: A high-level image representation for scene classification and semantic feature sparsification. *NIPS*, 2010.

[8] S. McCann and D. G. Lowe. Local naive bayes nearest neighbor for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3650–3656. IEEE, 2012.

[9] M. Norouzi, A. Punjani, and D. J. Fleet. Fast search in hamming space with multi-index hashing. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3108–3115. IEEE, 2012.

[10] A. Oliva and A. Torralba. Building the gist of a scene: The role of global image features in recognition. *Progress in brain research*, 155:23–36, 2006.

[11] Oliva, Aude and Torralba, Antonio. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *International Journal of Computer Vision*, 42(3):145–175, May 2001.

[12] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, pages 1–8, 2008.

[13] Torralba, A., Fergus, R., and Freeman, W. T. 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, November 2008.

[14] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, pages 3485–3492, 2010.